Roll no – 391

ASSIGNMENT NO - 4

Name – Neha Dattatray Bhoite
Sub - SPOS

1)Least Recently Used Algorithm -

```
def lru_page_replacement(reference_string, num_frames):
    frame = []  # To hold current pages in frames
    hits = 0
    faults = 0
    mem_layout = []  # To track memory layout at each step

    for page in reference_string:
        if page in frame:  # Page hit
            hits += 1
        else:  # Page fault
            faults += 1
            if len(frame) < num_frames:
                frame.append(page)  # Add page if there's space
            else:
                # Remove the least recently used page
                lru_page = frame.pop(0)
                frame.append(page)  # Add the new page

        mem_layout.append(frame.copy())  # Store current memory layout
return hits, faults, mem_layout

# Example usage
if __name__ == "__main__":
    reference_string = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3]
    num_frames = 3
    hits, faults, mem_layout = lru_page_replacement(reference_string, num_frames)

    # Print memory layout at each step
    print("Memory Layout:")
    for i, layout in enumerate(mem_layout):
        print(f"Step {i + 1}: {layout}")
    # Print the results
    print(f"\nThe number of Hits: {hits}")
    print(f"The number of Faults: {faults}")
    print(f"Hit Ratio: {hits / len(reference_string):.2f}")
```

Output-

```
Memory Layout:
Step 1: [7]
Step 2: [7, 0]
Step 3: [7, 0, 1]
Step 4: [0, 1, 2]
Step 5: [1, 2, 0]
Step 6: [2, 0, 3]
Step 7: [0, 3]
Step 8: [3, 4]
Step 9: [4, 2]
Step 10: [2, 3]
Step 11: [3, 0]
Step 12: [0, 3]

The number of Hits: 5
The number of Faults: 7
Hit Ratio: 0.42
```

2) Optimal page replacement-

```
def optimal_page_replacement(reference_string, num_frames):
    frames = []  # To hold current pages in frames
    hits = 0
    faults = 0
    mem_layout = []  # To track memory layout at each step

    for i, page in enumerate(reference_string):
        if page in frames:  # Page hit
            hits += 1
        else:  # Page fault
            faults += 1
        if len(frames) < num_frames:
            frames.append(page)  # Add page if there's space
        else:
            # Find the page to replace using the optimal strategy
            # We need to find the page that will not be used for the longest time in the future
            future_uses = {frame: float('inf') for frame in frames}
            for j in range(i + 1, len(reference_string)):
                if reference_string[j] in future_uses and future_uses[reference_string[j]] ==
float('inf'):
                    future_uses[reference_string[j]] = j

            # Select the frame to replace
            page_to_replace = max(future_uses, key=future_uses.get)
            frames.remove(page_to_replace)
            frames.append(page)
    mem_layout.append(frames.copy())  # Store current memory layout
    return hits, faults, mem_layout

# Example usage
if __name__ == "__main__":
    reference_string = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3]
    num_frames = 3
  hits, faults, mem_layout = optimal_page_replacement(reference_string, num_frames)

    # Print memory layout at each step
    print("Memory Layout:")
    for i, layout in enumerate(mem_layout):
        print(f"Step {i + 1}: {layout}")

    # Print the results
    print(f"\nThe number of Hits: {hits}")
    print(f"The number of Faults: {faults}")
    print(f"Hit Ratio: {hits / len(reference_string):.2f}")
```

Output-

```
Memory Layout:
Step 1: [7]
Step 2: [7, 0]
Step 3: [7, 0, 1]
Step 4: [2, 0, 1]
Step 5: [2, 0, 3]
Step 6: [0, 3]
Step 7: [3, 4]
Step 8: [4, 2]
Step 9: [4, 3]
Step 10: [3, 0]
Step 11: [0, 3]
Step 12: [0, 3]

The number of Hits: 5
The number of Faults: 7
Hit Ratio: 0.42
```

3) First In First Out Algorithm -

```
def fifo_page_replacement(reference_string, num_frames):
    frame = [-1] * num_frames  # Initialize the frames with -1
    pointer = 0  # Pointer for the next frame to be replaced
    hits = 0  # Count of page hits
    faults = 0  # Count of page faults
    mem_layout = []  # To track memory layout at each step

    for page in reference_string:
        if page in frame:  # Check if page is already in frame
            hits += 1
        else:
            faults += 1
            frame[pointer] = page  # Replace the page in the frame
            pointer = (pointer + 1) % num_frames  # Move pointer to the next frame

        mem_layout.append(frame.copy())  # Store the current memory layout

    return hits, faults, mem_layout

# Example usage
```

```
if __name__ == "__main__":
    reference_string = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3]
    num_frames = 3

    hits, faults, mem_layout = fifo_page_replacement(reference_string, num_frames)

    # Print memory layout at each step
    print("Memory Layout:")
    for i, layout in enumerate(mem_layout):
        print(f"Step {i + 1}: {layout}")

    # Print the results
    print(f"\nThe number of Hits: {hits}")
    print(f"The number of Faults: {faults}")
    print(f"Hit Ratio: {hits / len(reference_string):.2f}")
```

Output-

```
Memory Layout:
Step 1: [7, -1, -1]
Step 2: [7, 0, -1]
Step 3: [7, 0, 1]
Step 4: [2, 0, 1]
Step 5: [2, 0, 1]
Step 6: [2, 3, 1]
Step 7: [2, 3, 0]
Step 8: [4, 3, 0]
Step 9: [4, 2, 0]
Step 10: [4, 2, 3]
Step 11: [0, 2, 3]
Step 12: [0, 2, 3]

The number of Hits: 2
The number of Faults: 10
Hit Ratio: 0.17

=== Code Execution Successful ===
```